LAMPS
Internet-Draft
Intended status: Standards Track
Expires: 30 November 2023

M. Ounsworth
J. Gray
Entrust
M. Pala
CableLabs
J. Klaussner
D-Trust GmbH
29 May 2023

Composite Public and Private Keys For Use In Internet PKI
draft-ounsworth-pq-composite-keys-05

Abstract

The migration to post-quantum cryptography is unique in the history of modern digital cryptography in that neither the old outgoing nor the new incoming algorithms are fully trusted to protect data for the required data lifetimes. The outgoing algorithms, such as RSA and elliptic curve, may fall to quantum cryptalanysis, while the incoming post-quantum algorithms face uncertainty about both the underlying mathematics as well as hardware and software implementations that have not had sufficient maturing time to rule out classical cryptanalytic attacks and implementation bugs.

Cautious implementers may wish to layer cryptographic algorithms such that an attacker would need to break all of them in order to compromise the data being protected using either a Post-Quantum / Traditional Hybrid, Post-Quantum / Post-Quantum Hybrid, or combinations thereof. This document, and its companions, defines a specific instantiation of hybrid paradigm called "composite" where multiple cryptographic algorithms are combined to form a single key, signature, or key encapsulation mechanism (KEM) such that they can be treated as a single atomic object at the protocol level.

This document defines the structures CompositePublicKey and CompositePrivateKey, which are sequences of the respective structure for each component algorithm. Explicit pairings of algorithms are defined which should meet most Internet needs.

This document is intended to be coupled with corresponding documents that define the structure and semantics of composite signatures and encryption, such as [I-D.ounsworth-pq-composite-sigs] and [I-D.ounsworth-pq-composite-kem].

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ounsworth-pq-composite-keys/.

Discussion of this document takes place on the Limited Additional Mechanisms for PKIX and SMIME (lamps) Working Group mailing list (mailto:spasm@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/spasm/. Subscribe at https://www.ietf.org/mailman/listinfo/spasm/.

Source for this draft and an issue tracker can be found at https://github.com/EntrustCorporation/draft-ounsworth-pq-composite-keys.

Status of This Memo

Copyright Notice

Table of Contents

1. Changes in version -05

* Removed SPHINCS+ hybrids.

* Removed all references to generic composite.

* Added selection criteria note about requesting new explicit combinations.

2. Introduction

During the transition to post-quantum cryptography (PQ or PQC), there will be uncertainty as to the strength of cryptographic algorithms; we will no longer fully trust traditional cryptography such as RSA, Diffie-Hellman, DSA and their elliptic curve variants, but we may also not fully trust their post-quantum replacements until further time has passed to allow additional scrutiny and the discovery of implementation bugs. Unlike previous cryptographic algorithm migrations, the choice of when to migrate and which algorithms to migrate to, is not so clear. Even after the migration period, it may be advantageous for an entity's cryptographic identity to be composed of multiple public-key algorithms by using a Post-Quantum/Traditional (PQ/T) or Post-Quantum/Post-Quantum (PQ/PQ) Hybrid scheme.

The transition to PQC will face two challenges:

* Algorithm strength uncertainty: During the transition period, some post-quantum signature and encryption algorithms will not be fully trusted, while also the trust in legacy public key algorithms will start to erode. A relying party may learn some time after deployment that a public key algorithm has become untrustworthy, but in the interim, they may not know which algorithm an adversary has compromised.

* Migration: During the transition period, systems will require mechanisms that allow for staged migrations from fully traditional to fully post-quantum-aware cryptography.

This document provides the composite mechanism, which is a specific instantiation of the PQ/T hybrid paradigm to address algorithm strength uncertainty concerns by providing formats for encoding multiple public key and private key values into existing public key and private key fields. Backwards compatibility is not directly addressed via the composite mechanisms defined in the document, but some notes on how it can be obtained can be found in Appendix C.2. Other hybrid public key, signature, and KEM mechanisms exist, notably [I-D.guthrie-ipsecme-ikev2-hybrid-auth] and [I-D.truskovsky-lamps-pq-hybrid-x509] / [itu-t-x509-2019], which have their security and ease of migration properties discussed in more detail in Appendix C.2.

This document only specifies key formats; usage of these keys are covered in the corresponding composite signatures [I-D.ounsworth-pq-composite-sigs] and composite KEM [I-D.ounsworth-pq-composite-kem] specifications.

This document is intended for general applicability anywhere that keys are used within PKIX or CMS structures.

2.1. Algorithm Selection Criteria

The composite algorithm combinations defined in this document were chosen according to the following guidelines:

1. A single RSA combination is provided (but RSA modulus size not mandated), matched with NIST PQC Level 3 algorithms.

2. Elliptic curve algorithms are provided with combinations on each of the NIST [RFC6090], Brainpool [RFC5639], and Edwards [RFC7748] curves. NIST PQC Levels 1 - 3 algorithms are matched with 256-bit curves, while NIST levels 4 - 5 are matched with 384-bit elliptic curves. This provides a balance between matching classical security levels of post-quantum and traditional algorithms, and also selecting elliptic curves which already have wide adoption.

3. NIST level 1 candidates (Falcon512 and Kyber512) are provided, matched with 256-bit elliptic curves, intended for constrained use cases. The authors wish to note that although all the composite structures defined in this and the companion documents [I-D.ounsworth-pq-composite-sigs] and [I-D.ounsworth-pq-composite-kem] specifications are defined in such a way as to easily allow 3 or more component algorithms, it was decided to only specify explicit pairs. This also does not preclude future specification of explicit combinations with three or more components.

To maximize interoperability, use of the specific algorithm combinations specified in this document is encouraged. If other combinations are needed, a separate specification should be submitted to the IETF LAMPS working group. To ease implementation, these specifications are encouraged to follow the construction pattern of the algorithms specified in this document.

2.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document is consistent with all terminology from [I-D.driscoll-pqt-hybrid-terminology].

In addition, the following terms are used in this document:

BER: Basic Encoding Rules (BER) as defined in [X.690].

CLIENT: Any software that is making use of a cryptographic key. This includes a signer, verifier, encrypter, decrypter.

DER: Distinguished Encoding Rules as defined in [X.690].

PKI: Public Key Infrastructure, as defined in [RFC5280].

3. Composite Key Structures

In order to represent public keys and private keys that are composed of multiple algorithms, we define encodings consisting of a sequence of public key or private key primitives (aka "components") such that these structures can be used directly in existing public key fields such as those found in PKCS#10 [RFC2986], CMP [RFC4210], X.509 [RFC5280], CMS [RFC5652], and the Trust Anchor Format [RFC5914].

[I-D.driscoll-pqt-hybrid-terminology] defines composites as:

> _Composite Cryptographic Element_: A cryptographic element that incorporates multiple component cryptographic elements of the same type in a multi-algorithm scheme.

Composite keys as defined here follow this definition and should be regarded as a single key that performs a single cryptographic operation such key generation, signing, verifying, encapsulating, or decapsulating -- using its encapsulated sequence of component keys as if it was a single key. This generally means that the complexity of combining algorithms can and should be ignored by application and protocol layers and deferred to the cryptographic library layer.

3.1. pk-Composite

The following ASN.1 Information Object Class is a template to be used in defining all composite key types, with suitable replacements for the ASN.1 identifier pk-Composite and the OID id-composite-key as appropriate. See the ASN.1 Module in Section 5 for parmeterized as well as signature and KEM versions.

```
pk-Composite PUBLIC-KEY ::= {
    id id-composite-key
    KeyValue CompositePublicKey
    Params ARE ABSENT
    PrivateKey CompositePrivateKey
}
```

keyUsage is omitted here because composites may be formed for keys of any type, provided that any key usage specified MUST apply to all component keys. Composites MAY NOT be used to combine key types, for example to make a "dual-usage" key by combining a signing key with a KEM key.

3.2. CompositePublicKey

Composite public key data is represented by the following structure:

```
CompositePublicKey ::= SEQUENCE SIZE (2..MAX) OF SubjectPublicKeyInfo
```

A composite key MUST contain at least two component public keys. When the composite key is used in conjunction with an explicit composite algorithm identifier defined under section Section 4, the order of the component keys is determined by that algorithm identifier's definition.

A CompositePublicKey MUST NOT contain a component public key which itself describes a composite key; i.e. recursive CompositePublicKeys are not allowed. The purpose is a general reduction in complexity by not needing to consider nested key types.

Each element of a CompositePublicKey is a SubjectPublicKeyInfo object encoding a component public key. Each component SubjectPublicKeyInfo SHALL contain an AlgorithmIdentifier OID which identifies the public key type and parameters for the public key contained within it. See Section 4 for specific algorithms defined in this document.

When the CompositePublicKey must be provided in octet string or bit string format, the data structure is encoded as specified in Section 3.5.

3.2.1. Key Usage

Protocols such as X.509 [RFC5280] that specify a key usage along with the public key. For composite keys, a single key usage is specified for the entire public key and it MUST apply to all component keys. For example if a composite key is marked with a key usage of digitalSignature, then all component keys MUST be capable of producing digital signatures and handled with policies appropriate for digital signature keys. The composite mechanism MUST NOT be used to implement mixed-usage keys, for example, where a digitalSignature and a keyEncipherment key are combined together into a single composite key.

Specifications of explicit composite key types must specify allowable key usages for that type based on the types of the components.

3.2.2. Component Matching

Many cryptographic libraries will require treating each component key independently and thus expect a full SubjectPublicKeyInfo for each component at some layer of the software stack. This left two design choices: either we carry full SPKI for each component within the CompositePublicKey, or we compress it by only carrying the raw key bytes and force implementations to carry OID and parameter mapping tables to be able to reconstruct component SPKIs.

The authors decided to carry the full SPKIs in order to lessen the implementation complexity at the expense of a small amount of redundant data to transmit.

This design choice has a non-obvious security risk in that the algorithm carried within each component SPKI is redundant information which MUST match -- and can be inferred from -- the specification of the explicit algorithm.

Security consideration: Implementations SHOULD check that the component AlgorithmIdentifier OIDs and parameters match those expected by the definition of the explicit algorithm. Implementations SHOULD first parse a component's SubjectPublicKeyInfo.algorithm, and ensure that it matches what is expected for that position in the explicit key, and then proceed to parse the SubjectPublicKeyInfo.subjectPublicKey. This is to reduce the attack surface associated with parsing the public key data of an unexpected key type, or worse; to parse and use a key which does not match the explicit algorithm definition. Similar checks SHOULD be done when handling the corresponding private key.

3.3. CompositePrivateKey

This section provides an encoding for composite private keys intended for PKIX protocols and other applications that require an interoperable format for transmitting private keys, such as PKCS #12 [RFC7292] or CMP / CRMF [RFC4210], [RFC4211]. It is not intended to dictate a storage format in implementations not requiring interoperability of private key formats.

In some cases the private keys that comprise a composite key may not be represented in a single structure or even be contained in a single cryptographic module. The establishment of correspondence between public keys in a CompositePublicKey and private keys not represented in a single composite structure is beyond the scope of this document.

The composite private key data is represented by the following structure:

```
CompositePrivateKey ::= SEQUENCE SIZE (2..MAX) OF OneAsymmetricKey
```

Each element is a OneAsymmetricKey [RFC5958] object for a component private key.

The parameters field MUST be absent.

A CompositePrivateKey MUST contain at least two component private keys, and the order of the component keys is the same as the order defined in Section 3.2 for the components of CompositePublicKey.

3.4. As a PrivateKeyInfo or OneAsymmetricKey

A CompositePrivateKey can be stored in a OneAsymmetricKey structure (version 1 of which is also known as PrivateKeyInfo) [RFC5958]. When this is done, the privateKeyAlgorithm field SHALL be set to the corresponding composite algorithm identifier defined according to Section 4, the privateKey field SHALL contain the CompositePrivateKey, and the publicKey field MUST NOT be present. Associated public key material MAY be present in the CompositePrivateKey.

3.5. Encoding Rules

Many protocol specifications will require that the composite public key and composite private key data structures be represented by an octet string or bit string.

When an octet string is required, the DER encoding of the composite data structure SHALL be used directly.

CompositePublicKeyOs ::= OCTET STRING (CONTAINING CompositePublicKey ENCODED BY der)

When a bit string is required, the octets of the DER encoded composite data structure SHALL be used as the bits of the bit string, with the most significant bit of the first octet becoming the first bit, and so on, ending with the least significant bit of the last octet becoming the last bit of the bit string.

CompositePublicKeyBs ::= BIT STRING (CONTAINING CompositePublicKey ENCODED BY der)

4. Algorithm Identifiers

This section defines algorithm identifiers, component algorithms and their ordering for composite combinations. The combinations registered in this section are intended to strike a balance between the overall number of combinations ("the combinatorial explosion problem"), while also covering the needs of a wide range of protocols, applications, and regulatory environments in which X.509-based technologies are used.

This section is not intended to be exhaustive and other authors may define OIDs for new combinations so long as they are compatible with the structures and processes defined in this and the companion signature and encryption documents.

4.1. Signature public key types

This table summarizes the list of explicit composite Signature algorithms by the key and signature OID and the two component algorithms which make up the explicit composite algorithm, as obtained by applying the selection criteria in section Section 2.1. These are denoted by First Signature Alg, and Second Signature Alg.

The OID referenced are TBD and MUST be used only for prototyping and replaced with the final IANA-assigned OIDS. The following prefix is used for each: replace <CompSig> with the String "2.16.840.1.114027.80.5.1"

Therefore <CompSig>.1 is equal to 2.16.840.1.114027.80.5.1.1

Note that a single OID is used for both the key type and the signature algorithm; ie there is a one-to-one correspondance between key types and signature algorithms, hence why these key type names contain more information than they strictly need to define a key type.

Composite Signature Key Type	OID	First Key Type	Second Key Type
id-Dilithium3-RSA-PSS	<CompSig>.14	Dilithium3	RSASSA-PSS
id-Dilithium3-RSA-PKCS15-SHA256	<CompSig>.1	Dilithium3	RSAES-PKCS-v1_5
id-Dilithium3-ECDSA-P256-SHA256	<CompSig>.2	Dilithium3	EC-P256
id-Dilithium3-ECDSA-brainpoolP256r1-SHA256	<CompSig>.3	Dilithium3	EC-brainpoolP256r1
id-Dilithium3-Ed25519	<CompSig>.4	Dilithium3	Ed25519
id-Dilithium5-ECDSA-P384-SHA384	<CompSig>.5	Dilithium5	EC-P384
id-Dilithium5-ECDSA-brainpoolP384r1-SHA384	<CompSig>.6	Dilithium5	EC-brainpoolP384r1
id-Dilithium5-Ed448	<CompSig>.7	Dilithium5	Ed448
id-Falcon512-ECDSA-P256-SHA256	<CompSig>.8	Falcon512	EC-P256
id-Falcon512-ECDSA-brainpoolP256r1-SHA256	<CompSig>.9	Falcon512	EC-brainpoolP256r1
id-Falcon512-Ed25519	<CompSig>.10	Falcon512	Ed25519

Table 1

The table above contains everything needed to implement the listed explicit composite algorithms. See the ASN.1 module in section Section 5 for the explicit definitions of the above Composite signature algorithms.

Full specifications for the referenced algorithms can be found as follows:

* _Dilithium_: [I-D.ietf-lamps-dilithium-certificates]

* _EC_: [RFC5480]

  - _EC-P256_: AlgorithmIdentifier.parameters MUST be secp256r1 as defined in [RFC5480].

  - _EC-brainpoolP256r1_: AlgorithmIdentifier.parameters MUST be brainpoolP256r1 as defined in [RFC5639].

  - _EC-P384_: AlgorithmIdentifier.parameters MUST be secp384r1 as defined in [RFC5480].

  - _EC-brainpoolP384r1_: AlgorithmIdentifier.parameters MUST be brainpoolP384r1 as defined in [RFC5639].

* _Ed25519 / Ed448_: [RFC8410]

* _Falcon_: TBD

* _RSAES-PKCS-v1_5_: [RFC8017]

* _RSASSA-PSS_: [RFC8017]

The intended application for the key is indicated in the keyUsage certificate extension; see Section 4.2.1.3 of [RFC5280]. If the keyUsage extension is present in a certificate that indicates signature public key types above in the SubjectPublicKeyInfo, then the at least one of following MUST be present:

```
digitalSignature; or
nonRepudiation; or
keyCertSign; or
cRLSign.
```

Requirements about the keyUsage extension bits defined in [RFC5280] still apply.

4.2. KEM public key types

This table summarizes the list of explicit composite Signature algorithms by the key and signature OID and the two component algorithms which make up the explicit composite algorithm. These are denoted by First Signature Alg, and Second Signature Alg.

The OID referenced are TBD and MUST be used only for prototyping and replaced with the final IANA-assigned OIDS. The following prefix is used for each: replace <CompKEM> with the String "2.16.840.1.114027.80.5.2"

Therefore <CompKEM>.1 is equal to 2.16.840.1.114027.80.5.2.1.

Note that a single OID is used for both the key type and the KEM algorithm; ie there is a one-to-one correspondance between key types and KEM algorithms, hence why these key type names contain more information than they strictly need to define a key type.

Composite KEM Key Type	OID	First Key Type	Second Key Type
id-Kyber512-ECDH-P256-KMAC128	<CompKEM>.1	Kyber512	EC-P256
id-Kyber512-ECDH-brainpoolP256r1-KMAC128	<CompKEM>.2	Kyber512	EC-brainpoolP256r1
id-Kyber512-X25519-KMAC128	<CompKEM>.3	Kyber512	X25519
id-Kyber768-RSA-KMAC256	<CompKEM>.4	Kyber768	RSA-KEM
id-Kyber768-ECDH-P256-KMAC256	<CompKEM>.5	Kyber768	EC-P256
id-Kyber768-ECDH-brainpoolP256r1-KMAC256	<CompKEM>.6	Kyber768	EC-brainpoolP256r1
id-Kyber768-X25519-KMAC256	<CompKEM>.7	Kyber768	X25519
id-Kyber1024-ECDH-P384-KMAC256	<CompKEM>.8	Kyber1024	EC-P384
id-Kyber1024-ECDH-brainpoolP384r1-KMAC256	<CompKEM>.9	Kyber1024	EC-brainpoolP384r1
id-Kyber1024-X448-KMAC256	<CompKEM>.10	Kyber1024	X448

Table 2: Composite KEM key types

The table above contains everything needed to implement the listed explicit composite algorithms. See the ASN.1 module in section Section 5 for the explicit definitions of the above Composite signature algorithms.

Full specifications for the referenced algorithms can be found as follows:

* _EC_: [RFC5480]

  - _EC-P256_: AlgorithmIdentifier.parameters within the component SKPI belonging to the EC key MUST be secp256r1 as defined in [RFC5480].

  - _EC-brainpoolP256r1_: AlgorithmIdentifier.parameters within the component SKPI belonging to the EC key MUST be brainpoolP256r1 as defined in [RFC5639].

  - _EC-P384_: AlgorithmIdentifier.parameters within the component SKPI belonging to the EC key MUST be secp384r1 as defined in [RFC5480].

  - _EC-brainpoolP384r1_: AlgorithmIdentifier.parameters within the component SKPI belonging to the EC key MUST be brainpoolP384r1 as defined in [RFC5639].

* _Kyber_: [I-D.ietf-lamps-kyber-certificates]

* _RSA-KEM_: [RFC5990]

* _X25519 / X448_: [RFC8410]

Note: the inclusion of a hash function is so that these algorithm identifiers can double as both key types and KEM algorithms.

The intended application for the key is indicated in the keyUsage certificate extension; see Section 4.2.1.3 of [RFC5280]. If the keyUsage extension is present in a certificate that indicates any of the KEM public key types above in the SubjectPublicKeyInfo, then the following MUST be present:

  keyEncipherment

Requirements about the keyUsage extension bits defined in [RFC5280] still apply.

5. ASN.1 Module

<CODE STARTS>

!!Composite-Keys-2023.asn

<CODE ENDS>

6. IANA Considerations

All sorts of OIDs in the ASN.1 module. Too many to list here (sorry).

This document registers the following in the SMI "Security for PKIX Algorithms (1.3.6.1.5.5.7.6)" registry:

TODO

7. Security Considerations

7.1. Reuse of keys in a Composite public key

There is an additional security consideration that some use cases such as signatures remain secure against downgrade attacks if and only if component keys are never used outside of their composite context and therefore it is RECOMMENDED that component keys in a composite key are not to be re-used in other contexts. In particular, the components of a composite key SHOULD NOT also appear in single-key certificates. This is particularly relevant for protocols that use composite keys in a logical AND mode since the appearance of the same component keys in single-key contexts undermines the binding of the component keys into a single composite key by allowing messages signed in a multi-key AND mode to be presented as if they were signed in a single key mode in what is known as a "stripping attack".

7.2. Key mismatch in explicit composite

This security consideration copied from Section 3.2.2.

Implementations SHOULD check that the component AlgorithmIdentifier OIDs and parameters match those expected by the definition of the explicit algorithm. Implementations SHOULD first parse a component's SubjectPublicKeyInfo.algorithm, and ensure that it matches what is expected for that position in the explicit key, and then proceed to parse the SubjectPublicKeyInfo.subjectPublicKey. This is to reduce the attack surface associated with parsing the public key data of an unexpected key type, or worse; to parse and use a key which does not match the explicit algorithm definition. Similar checks SHOULD be done when handling the corresponding private key.

7.3. Policy for Deprecated and Acceptable Algorithms

Traditionally, a public key, certificate, or signature contains a single cryptographic algorithm. If and when an algorithm becomes deprecated (for example, RSA-512, or SHA1), it is obvious that clients performing signature verification or encryption operations should be updated to fail to validate or refuse to encrypt for these algorithms.

In the composite model this is less obvious since implementers may decide that certain cryptographic algorithms have complementary security properties and are acceptable in combination even though one or both algorithms are deprecated for individual use. As such, a single composite public key, certificate, signature, or ciphertext MAY contain a mixture of deprecated and non-deprecated algorithms.

Specifying behaviour in these cases is beyond the scope of this document, but should be considered by implementers and potentially in additional standards.

EDNOTE: Max had proposed a CRL mechanism to accomplish this, which could be revived if necessary.

7.4. Protection of Private Keys

Structures described in this document do not protect private keys in any way unless combined with a security protocol or encryption properties of the objects (if any) where the CompositePrivateKey is used.

Protection of the private keys is vital to public key cryptography. The consequences of disclosure depend on the purpose of the private key. If a private key is used for signature, then the disclosure allows unauthorized signing. If a private key is used for key management, then disclosure allows unauthorized parties to access the managed keying material. The encryption algorithm used in the encryption process must be at least as 'strong' as the key it is protecting.

7.5. Checking for Compromised Key Reuse

Certification Authority (CA) implementations need to be careful when checking for compromised key reuse, for example as required by WebTrust regulations; when checking for compromised keys, you MUST unpack the CompositePublicKey structure and compare individual component keys. In other words, for the purposes of key reuse checks, the composite public key structures need to be un-packed so that primitive keys are being compared. For example if the composite

key {RSA1, PQ1} is revoked for key compromise, then the keys RSA1 and PQ1 need to be individually considered revoked. If the composite key {RSA1, PQ2} is submitted for certification, it SHOULD be rejected because the key RSA1 was previously declared compromised even though the key PQ2 is unique.

8. References

8.1. Normative References

[I-D.ietf-lamps-dilithium-certificates]
Massimo, J., Kampanakis, P., Turner, S., and B. Westerbaan, "Internet X.509 Public Key Infrastructure: Algorithm Identifiers for Dilithium", Work in Progress, Internet-Draft, draft-ietf-lamps-dilithium-certificates-00, 29 September 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-dilithium-certificates-00>.

[I-D.ietf-lamps-kyber-certificates]
Turner, S., Kampanakis, P., Massimo, J., and B. Westerbaan, "Internet X.509 Public Key Infrastructure - Algorithm Identifiers for Kyber", Work in Progress, Internet-Draft, draft-ietf-lamps-kyber-certificates-00, 26 September 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-kyber-certificates-00>.

[I-D.ounsworth-pq-composite-kem]
Ounsworth, M. and J. Gray, "Composite KEM For Use In Internet PKI", Work in Progress, Internet-Draft, draft-ounsworth-pq-composite-kem-00, 11 July 2022, <https://datatracker.ietf.org/doc/html/draft-ounsworth-pq-composite-kem-00>.

[I-D.ounsworth-pq-composite-sigs]
Ounsworth, M. and M. Pala, "Composite Signatures For Use In Internet PKI", Work in Progress, Internet-Draft, draft-ounsworth-pq-composite-sigs-05, 12 July 2021, <https://datatracker.ietf.org/doc/html/draft-ounsworth-pq-composite-sigs-05>.

[RFC1421] Linn, J., "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures", RFC 1421, DOI 10.17487/RFC1421, February 1993, <https://www.rfc-editor.org/info/rfc1421>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <https://www.rfc-editor.org/info/rfc5480>.

[RFC5639] Lochter, M. and J. Merkle, "Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation", RFC 5639, DOI 10.17487/RFC5639, March 2010, <https://www.rfc-editor.org/info/rfc5639>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5914] Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor Format", RFC 5914, DOI 10.17487/RFC5914, June 2010, <https://www.rfc-editor.org/info/rfc5914>.

[RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <https://www.rfc-editor.org/info/rfc5958>.

[RFC5990] Randall, J., Kaliski, B., Brainard, J., and S. Turner, "Use of the RSA-KEM Key Transport Algorithm in the Cryptographic Message Syntax (CMS)", RFC 5990, DOI 10.17487/RFC5990, September 2010, <https://www.rfc-editor.org/info/rfc5990>.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, DOI 10.17487/RFC6090, February 2011, <https://www.rfc-editor.org/info/rfc6090>.

[RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/RFC7468, April 2015, <https://www.rfc-editor.org/info/rfc7468>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8410] Josefsson, S. and J. Schaad, "Algorithm Identifiers for Ed25519, Ed448, X25519, and X448 for Use in the Internet X.509 Public Key Infrastructure", RFC 8410, DOI 10.17487/RFC8410, August 2018, <https://www.rfc-editor.org/info/rfc8410>.

[RFC8411] Schaad, J. and R. Andrews, "IANA Registration for the Cryptographic Algorithm Object Identifier Range", RFC 8411, DOI 10.17487/RFC8411, August 2018, <https://www.rfc-editor.org/info/rfc8411>.

[X.690] ITU-T, "Information technology - ASN.1 encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ISO/IEC 8825-1:2015, November 2015.

8.2. Informative References

[codeSigningBRsv2.8] CAB Forum, "Baseline Requirements for the Issuance and Management of Publicly-Trusted Code Signing Certificates v2.8", May 2022, <https://cabforum.org/wp-content/uploads/Baseline-Requirements-for-the-Issuance-and-Management-of-Code-Signing.v2.8.pdf>.

[eIDAS2014]
"REGULATION (EU) No 910/2014 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC", July 2014, <https://ec.europa.eu/futurium/en/system/files/ged/eidas_regulation.pdf>.

[I-D.becker-guthrie-noncomposite-hybrid-auth]
Becker, A., Guthrie, R., and M. J. Jenkins, "Non-Composite Hybrid Authentication in PKIX and Applications to Internet Protocols", Work in Progress, Internet-Draft, draft-becker-guthrie-noncomposite-hybrid-auth-00, 22 March 2022, <https://datatracker.ietf.org/doc/html/draft-becker-guthrie-noncomposite-hybrid-auth-00>.

[I-D.driscoll-pqt-hybrid-terminology]
D, F., "Terminology for Post-Quantum Traditional Hybrid Schemes", Work in Progress, Internet-Draft, draft-driscoll-pqt-hybrid-terminology-01, 20 October 2022, <https://datatracker.ietf.org/doc/html/draft-driscoll-pqt-hybrid-terminology-01>.

[I-D.guthrie-ipsecme-ikev2-hybrid-auth]
Guthrie, R., "Hybrid Non-Composite Authentication in IKEv2", Work in Progress, Internet-Draft, draft-guthrie-ipsecme-ikev2-hybrid-auth-00, 25 March 2022, <https://datatracker.ietf.org/doc/html/draft-guthrie-ipsecme-ikev2-hybrid-auth-00>.

[I-D.truskovsky-lamps-pq-hybrid-x509]
Truskovsky, A., Van Geest, D., Fluhrer, S., Kampanakis, P., Ounsworth, M., and S. Mister, "Multiple Public-Key Algorithm X.509 Certificates", Work in Progress, Internet-Draft, draft-truskovsky-lamps-pq-hybrid-x509-01, 29 August 2018, <https://datatracker.ietf.org/doc/html/draft-truskovsky-lamps-pq-hybrid-x509-01>.

[itu-t-x509-2019]
ITU-T, "ITU-T X.509 The Directory: Public-key and attribute certificate frameworks", January 2019, <https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=X.509>.

[RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <https://www.rfc-editor.org/info/rfc4210>.

[RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, DOI 10.17487/RFC4211, September 2005, <https://www.rfc-editor.org/info/rfc4211>.

[RFC7292] Moriarty, K., Ed., Nystrom, M., Parkinson, S., Rusch, A., and M. Scott, "PKCS #12: Personal Information Exchange Syntax v1.1", RFC 7292, DOI 10.17487/RFC7292, July 2014, <https://www.rfc-editor.org/info/rfc7292>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/ Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

Appendix A. Work in Progress

A.1. Combiner modes (KofN)

For content commitment use-cases, such as legally-binding non-repudiation, the signer (whether it be a CA or an end entity) needs to be able to specify how its signature is to be interpreted and verified.

For now we have removed combiner modes (AND, OR, KofN) from this draft, but we are still discussing how to incorporate this for the cases where it is needed (maybe a X.509 v3 extension, or a signature algorithm param).

Appendix B. Samples

These samples are reproduced here for completeness, but are also available in github:

https://github.com/EntrustCorporation/draft-ounsworth-pq-composite-keys/tree/master/sampledata

TODO: move these to https://github.com/lamps-wg before publication

B.1. Explicit Composite Public Key Samples

B.1.1. id-Dilithium3-ECDSA-P256

This example uses the following OID as defined in Open Quantum Safe, which correspond to NIST Round3 candidates:

https://github.com/open-quantum-safe/oqs-provider/blob/main/ALGORITHMS.md

id-dilithium3_aes 1.3.6.1.4.1.2.267.11.6.5

A Dilithium3-ECDSA-P256 public key:

!!sampledata/current/id-Dilithium3-ECDSA-P256_pub.pem

The corresponding explicit private key is as follows. Note that the PQ key comes from OpenQuantumSafe-openssl and is in the {privatekey || publickey} concatenated format. This may cause interoperability issues with some clients, and also makes the private keys appear larger than they would be if generated by a non-openssl client.

!!sampledata/current/id-Dilithium3-ECDSA-P256_priv.pem

B.1.2. id-Dilithium3-RSA

This example uses the following OID as defined in Open Quantum Safe, which correspond to NIST Round3 candidates:

https://github.com/open-quantum-safe/oqs-provider/blob/main/ALGORITHMS.md

id-dilithium3_aes 1.3.6.1.4.1.2.267.11.6.5

A Dilithium3-RSA public key:

!!sampledata/current/id-Dilithium3-RSA_pub.pem

The corresponding explicit private key is as follows. Note that the PQ key comes from OpenQuantumSafe-openssl and is in the {privatekey || publickey} concatenated format. This may cause interoperability issues with some clients, and also makes the private keys appear larger than they would be if generated by a non-openssl client.

!!sampledata/current/id-Dilithium3-RSA_priv.pem

B.1.3. id-Falcon512-ECDSA-P256

This example uses the following OID as definid in Open Quantum Safe, which correspond to NIST Round3 candidates:

https://github.com/open-quantum-safe/oqs-provider/blob/main/ALGORITHMS.md

id-falcon512 1.3.9999.3.1

A Falcon512-ECDSA-P256 public key:

!!sampledata/current/id-Falcon512-ECDSA-P256_pub.pem

The corresponding explicit private key is as follows. Note that the PQ key comes from OpenQuantumSafe-openssl and is in the {privatekey || publickey} concatenated format. This may cause interoperability issues with some clients, and also makes the private keys appear larger than they would be if generated by a non-openssl client.

!!sampledata/current/id-Falcon512-ECDSA-P256_priv.pem

Appendix C. Implementation Considerations

This section addresses practical issues of how this draft affects other protocols and standards.

EDNOTE 10: Possible topics to address:

* The size of these certs and cert chains.
* In particular, implications for (large) composite keys / signatures / certs on the handshake stages of TLS and IKEv2.
* If a cert in the chain is a composite cert then does the whole chain need to be of composite Certs?

* We could also explain that the root CA cert does not have to be of the same algorithms. The root cert SHOULD NOT be transferred in the authentication exchange to save transport overhead and thus it can be different than the intermediate and leaf certs.

C.1. Textual encoding of Composite Private Keys

CompositePrivateKeys can be encoded to the Privacy-Enhanced Mail (PEM) [RFC1421] format by placing a CompositePrivateKey into the privateKey field of a PrivateKeyInfo (OneAsymmetricKey) object, and then applying the PEM encoding rules as defined in [RFC7468] section 10 and 11 for plaintext and encrypted private keys, respectively.

C.2. Backwards Compatibility

As noted in the introduction, the post-quantum cryptographic migration will face challenges in both ensuring cryptographic strength against adversaries of unknown capabilities, as well as providing ease of migration. The composite mechanisms defined in this document primarily address cryptographic strength, however this section contains notes on how backwards compatibility may be obtained.

The term "ease of migration" is used here to mean that existing systems can be gracefully transitioned to the new technology without requiring large service disruptions or expensive upgrades. The term "backwards compatibility" is used here to mean something more specific; that existing systems, as they are deployed today, can interoperate with the upgraded systems of the future.

These migration and interoperability concerns need to be thought about in the context of various types of protocols that make use of X.509 and PKIX with relation to public key objects, from online negotiated protocols such as TLS 1.3 [RFC8446] and IKEv2 [RFC7296], to non-negotiated asynchronous protocols such as S/MIME signed and encrypted email [RFC8551], document signing such as in the context of the European eIDAS regulations [eIDAS2014], and publicly trusted code signing [codeSigningBRsv2.8], as well as myriad other standardized and proprietary protocols and applications that leverage CMS [RFC5652] signed or encrypted structures.

C.2.1. OR modes

This document purposefully does not specify how clients are to combine component keys together to form a single cryptographic operation; this is left up to the specifications of signature and encryption algorithms that make use of the composite key type. One possible way to combine component keys is through an OR relation, or OR-like client policies for acceptable algorithm combinations, where senders and / or receivers are permitted to ignore some component keys. Some envisioned uses of this include environments where the client encounters a component key for which it does not possess a compatible algorithm implementation but wishes to proceed with the cryptographic operation using the subset of component keys for which it does have compatible implementations. Such a mechanism could be designed to provide ease of migration by allowing for composite keys to be distributed and used before all clients in the environment are fully upgraded, but it does not allow for full backwards compatibility since clients would at least need to be upgraded from their current state to be able to parse the composite structures.

C.2.2. Parallel PKIs

We present the term "Parallel PKI" to refer to the setup where a PKI end entity possesses two or more distinct public keys or certificates for the same key type (signature, key establishment, etc) for the same identity (name, SAN), but containing keys for different cryptographic algorithms. One could imagine a set of parallel PKIs where an existing PKI using legacy algorithms (RSA, ECC) is left operational during the post-quantum migration but is shadowed by one or more parallel PKIs using pure post quantum algorithms or composite algorithms (legacy and post-quantum). This concept contains strong overlap with other documented approaches, such as [I-D.becker-guthrie-noncomposite-hybrid-auth] and highlights the synergy between composite and non-composite hybrid approaches.

Equipped with a set of parallel public keys in this way, a client would have the flexibility to choose which public key(s) or certificate(s) to use in a given cryptographic operation.

For negotiated protocols, the client could choose which public key(s) or certificate(s) to use based on the negotiated algorithms, or could combine two of the public keys for example in a non-composite hybrid method such as [I-D.becker-guthrie-noncomposite-hybrid-auth] or [I-D.guthrie-ipsecme-ikev2-hybrid-auth]. Note that it is possible to use the signature algorithm defined in [I-D.ounsworth-pq-composite-sigs] as a way to carry the multiple signature values generated by a non-composite public mechanism in protocols where it is easier to support the composite signature

algorithms than to implement such a mechanism in the protocol itself. There is also nothing precluding a composite public key from being one of the components used within a non-composite authentication operation; this may lead to greater convenience in setting up parallel PKI hierarchies that need to service a range of clients implementing different styles of post-quantum migration strategies.

For non-negotiated protocols, the details for obtaining backwards compatibility will vary by protocol, but for example in CMS [RFC5652], the inclusion of multiple SignerInfo or RecipientInfo objects is often already treated as an OR relationship, so including one for each of the end entity's parallel PKI public keys would, in many cases, have the desired effect of allowing the receiver to choose one they are compatible with and ignore the others, thus achieving full backwards compatibility.

C.2.3. CATALYST certificates

CATALYST certificates, defined in [I-D.truskovsky-lamps-pq-hybrid-x509] and [itu-t-x509-2019] provides an alternative mechanism for placing multiple public keys and signatures into a certificate via the X.509v3 extensions subjectAltPublicKeyInfo, altSignatureAlgorithm, and altSignatureValue. [itu-t-x509-2019] specifies that only one of the keys is to be used at a time, so it is not in fact a hybrid mechinism in that it is not providing dual algorithm security; instead it is merely a migration mechanism. One could imagine obtaining dual algorithm security by using a CATALYST certificate in a mode other than that described in [itu-t-x509-2019] where both keys produce a signature and place them, for example, togeher in a CompositeSignatureValue.

CALATYST certificates appear to have a backwards compatibility advantage in that these non-critical extensions will be ignored by legacy clients, thus making the certificate verification seamlessly verifiable by legacy clients. However, at the protocol level, the certificate holder still needs to know which algorithm the peer wants it to use in the protocol-level message. CATALYST certificates also have the disadvantage of needing to transmit the large post-quantum keys, signatures or key exchange data even if the client will not use them. Thus while CATALYST certificates may be advantageous in some applications that use multiple algorithms but can only handle a single certificate, it is in general not clear that they offer any strong advantage over a multi-cert hybrid in terms of ease of migration, or over composite in terms of security.

Appendix D. Intellectual Property Considerations

The following IPR Disclosure relates to this draft:

https://datatracker.ietf.org/ipr/3588/

Appendix E. Contributors and Acknowledgements

This document incorporates contributions and comments from a large group of experts. The Editors would especially like to acknowledge the expertise and tireless dedication of the following people, who attended many long meetings and generated millions of bytes of electronic mail and VOIP traffic over the past year in pursuit of this document:

John Gray (Entrust),
Serge Mister (Entrust),
Scott Fluhrer (Cisco Systems),
Panos Kampanakis (Cisco Systems),
Daniel Van Geest (ISARA),
Tim Hollebeek (Digicert),
Klaus-Dieter Wirth (D-Trust),
Patrick Kelsey (Not for Radio LLC),
Anthony Hu (wolfSSL), and
Francois Rousseau.

We are grateful to all, including any contributors who may have been inadvertently omitted from this list.

This document borrows text from similar documents, including those referenced below. Thanks go to the authors of those documents. "Copying always makes things easier and less error prone" - [RFC8411].

E.1. Making contributions

Additional contributions to this draft are welcome. Please see the working copy of this draft at, as well as open issues at:

https://github.com/EntrustCorporation/draft-ounsworth-pq-composite-keys

Authors' Addresses

Mike Ounsworth
Entrust Limited
2500 Solandt Road -- Suite 100
Ottawa, Ontario K2K 3G5
Canada
Email: mike.ounsworth@entrust.com

John Gray
Entrust Limited
2500 Solandt Road -- Suite 100
Ottawa, Ontario K2K 3G5
Canada
Email: john.gray@entrust.com

Massimiliano Pala
CableLabs
Email: director@openca.org

Jan Klaussner
D-Trust GmbH
Kommandantenstr. 15
10969 Berlin
Germany
Email: jan.klaussner@d-trust.net